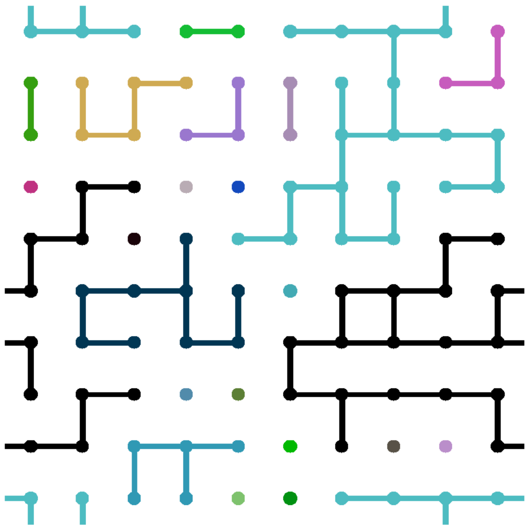
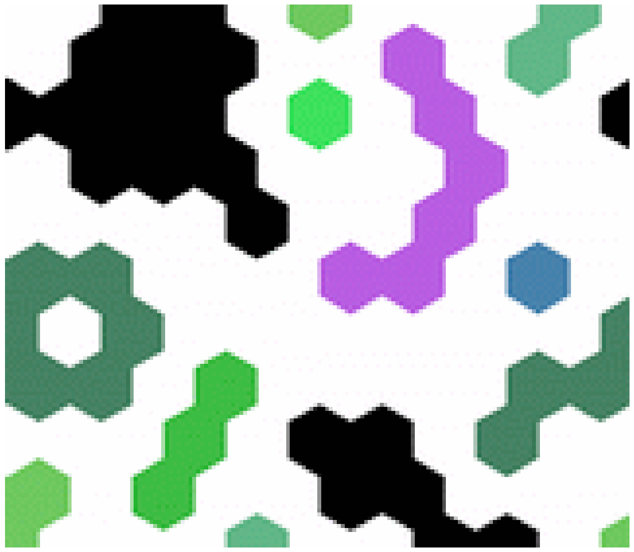
Percolation

Phys 682 / CIS 629: Computational Methods for Nonlinear Systems



bond percolation

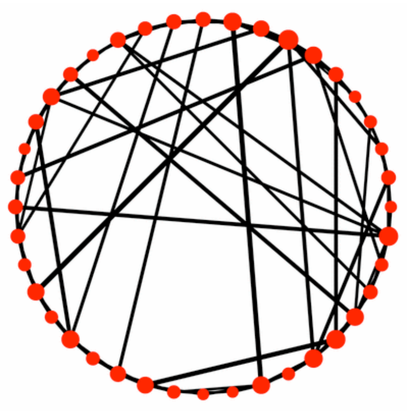


site percolation

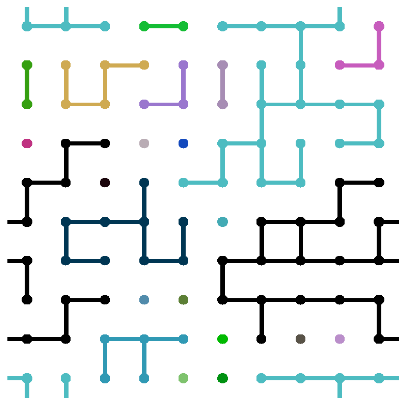
- Some applications
  - flow in porous media (e.g., pressure-driven flow in rock)
  - conductivity of disordered systems (e.g., random resistor networks)
  - forest fires
  - disease transmission in social networks
  - patterns of retinal activity

Networks, Nodes, Code Reuse & Dynamic Typing

- Many percolation simulations explicitly assume a grid of nodal values, but we're really just interested in an undirected graph.
- Can we reuse our UndirectedGraph code? What constitutes a node?
- In dynamically typed language (like Python), any object can be used as long its behavior is consistent with what is expected of it (e.g., a node can be used as a key in a dictionary).
- In a statically typed language (like C++ or Java), we would need to define a special `Node` base class to derive from.
- For percolation on a 2D lattice, want nodes as (i,j) index pairs.

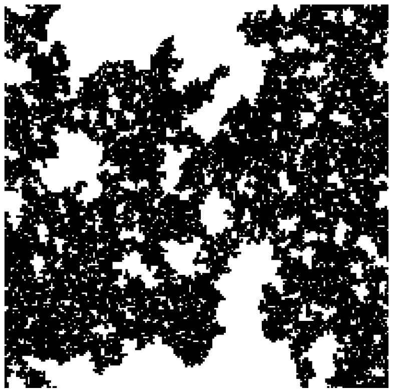
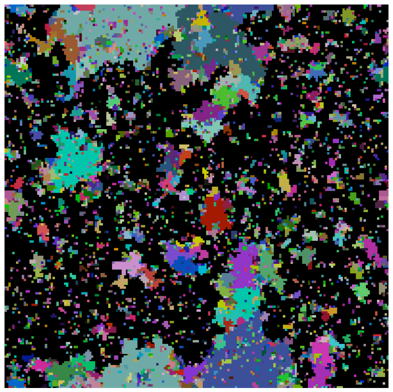
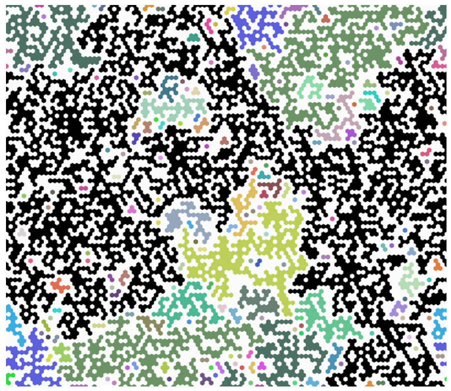


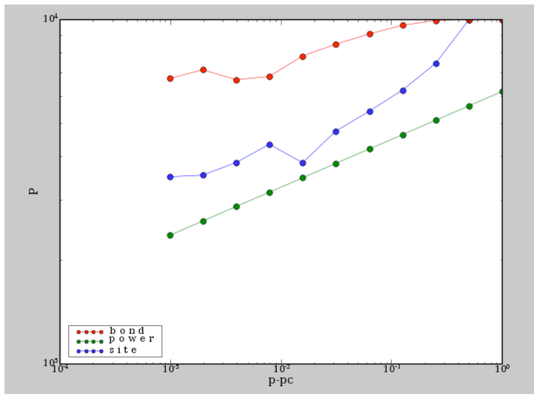
```
g = Networks.UndirectedGraph()
for i in range(L):
  for j in range(L):
     g.AddNode((i,j))
     if random.random() < p:
        g.AddEdge((i,j), ((i+1)%L,j))
     if random.random() < p:
        g.AddEdge((i,j), (i,(j+1)%L))
```



Power laws, correlation lengths, finite-size scaling & universality

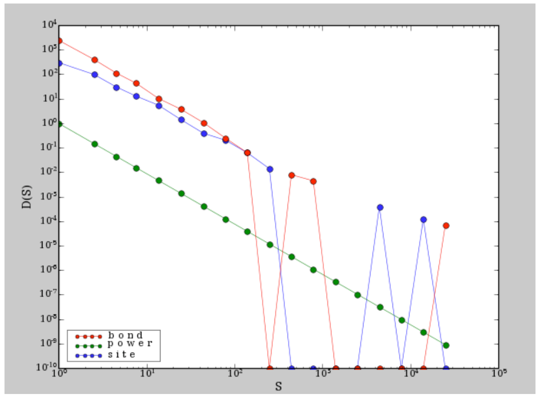
- Critical points imply scale invariance and power laws
- Phase transitions often involve a diverging correlation length $\xi \sim |p-p_c|^{-\nu}$
- Diverging correlation length constrained by finite system size → finite-size scaling
- Microscopically different systems can exhibit the same critical properties → universality



$$P(p) \sim (p-p_c)^{\beta}$$

probability of being in infinite cluster



$$D(s) \sim s^{-\tau}$$

cluster size distribution

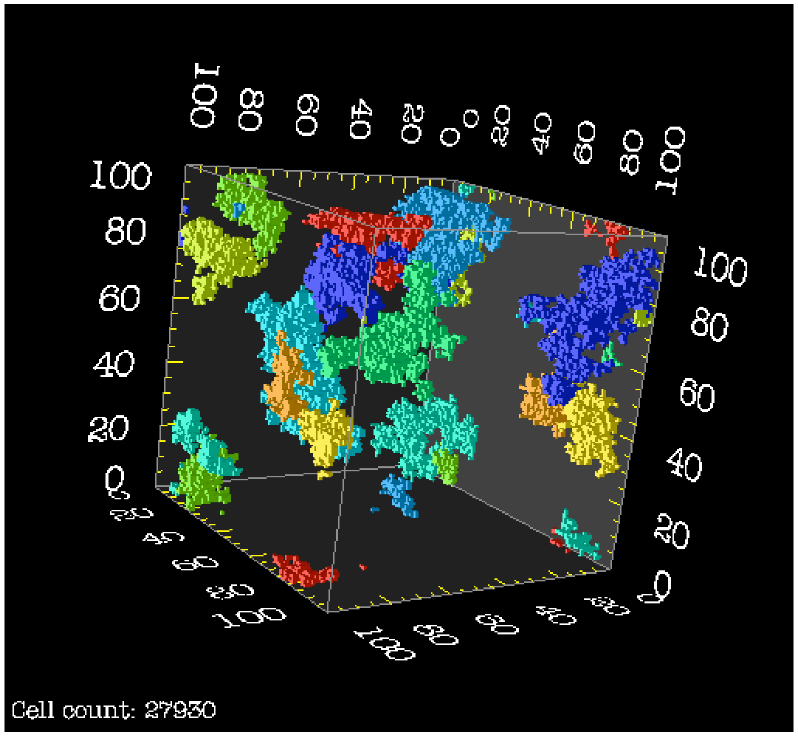
Three-dimensional bond percolation

Write new network generation code

```
g = Networks.UndirectedGraph()
for i in range(L):
  for j in range(L):
    for k in range(L):
      g.AddNode((i,j,k))
      if random.random() < p:
        g.AddEdge((i,j,k), ((i+1)%L,j,k))
      if random.random() < p:
        g.AddEdge((i,j,k), (i,(j+1)%L,k))
      if random.random() < p:
        g.AddEdge((i,j,k), (i,j,(k+1)%L))
```

Everything else (except visualization) works as for 2D percolation

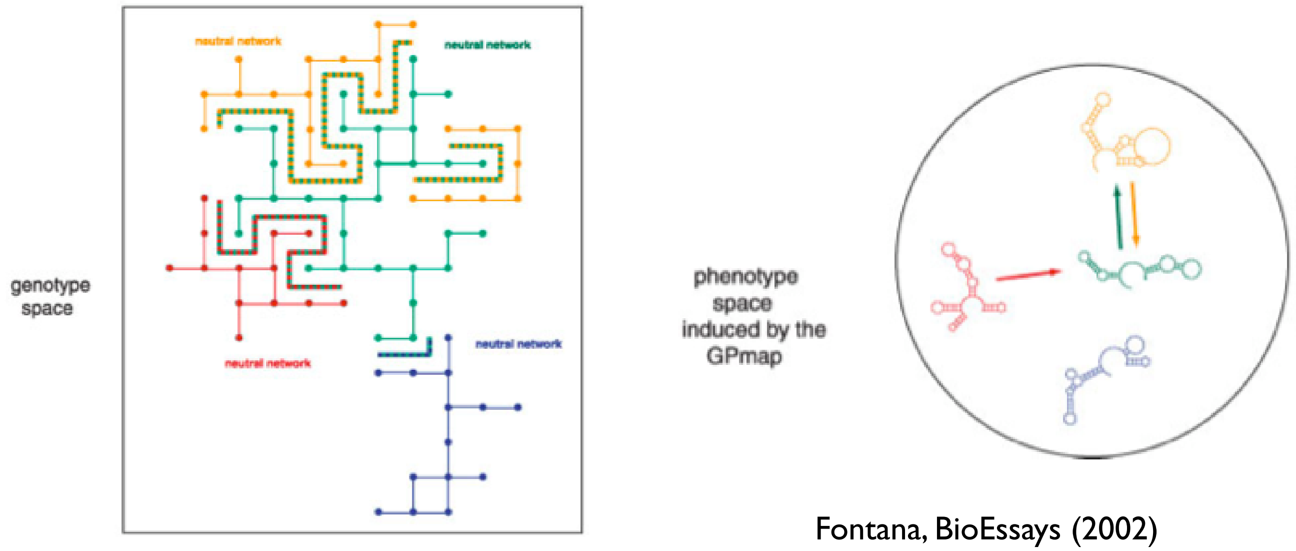
...or, one could percolate an existing network (by removing bonds with some probability) to understand its structure by seeing how it falls apart.



visualization in OpenDX
(by Chris Pelkie, CTC)

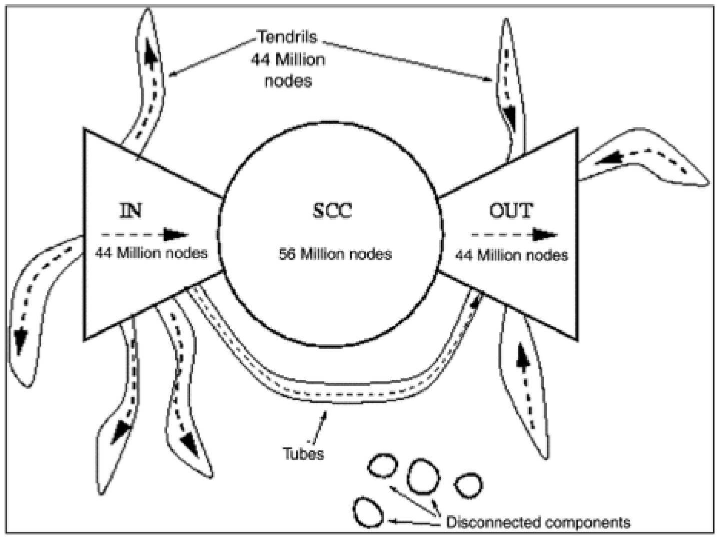
Connected components in networks

Neutral evolutionary networks in biology: what are the mutationally-connected networks of genotypes that produce the same phenotype?



Fontana, BioEssays (2002)

The bow-tie structure of the World Wide Web (SCC = strongly connected component in a directed graph)



Broder et al., Computer Networks (2000)